DATABASE MANAGEMENT SYSTEM

The present invention relates to database systems, and in particular to a database system which configures a data model in accordance with a hierarchical tree-like structure which enables fast and comprehensive data extraction and querying functions.

There are presently very many ways of constructing and maintaining database structures on computer systems. As is well known, the relational database is widely used. In a relational database, every entity in a data model has a number of attributes which may be accorded values selected either from discrete sets of values, or from within ranges of continuously variable values. All entity occurrences having the same attribute types are stored in a relation table, with each entity occurrence occupying a row, or tuple of the table having a field or element corresponding to each attribute. Each field of the row contains an alphanumeric value for the relevant attribute value. Separate tables are provided for different entities each having a different set of attributes.

The data model, or representation of the relationships between the different entities is provided both implicitly by the incidence of common attributes between the various relation tables, and also by imposing conditions on various attributes such as their identification as key fields.

In extracting data from the database, a query is formulated, in suitable programming language, which instructs the data processing system to scan selected attribute columns of specified tables for adherence to certain conditions, and to output, usually into an output table, the data in preselected attribute columns for each tuple or row of the scanned table or tables. The output table can then be browsed by the user on screen, or printed out.

A number of disadvantages present themselves with this technique. Queries must be formulated using particular query languages which must be learnt by the users. Although these are commonly interfaced with a "natural language" interface making their use easier for the non-expert user, certain rules and protocols must be understood.

A further disadvantage is that the queries are quite specific, and do not generally permit what we shall call "progressive browsing": that is to say, once a query has been formulated, the resulting output table is produced, and the information contained therein is fixed and limited to the scope of the original query. Further scanning of the output table is possible by formulating a further query to reduce the size of the output by imposing additional limitations on the ranges of values that an attribute may take, for example, but generally, for browsing through the database, a new query must be formulated each time to scan the appropriate parts of the database. In general (except where the selected attribute is the index field), in re-scanning the output table(s) to answer a "sub-query", the whole of the table or tables must be searched for adherence to the new selection criteria.

In processing a query, it is normally necessary to perform quite complex manipulations on the various tables involved in the query, which include joining or merging operations, and the temporary creation of intermediate tables to be used as the operands for subsequent parts of the query. Such operations naturally involve considerable processing power and time to carry out.

A further disadvantage is that the relational database must generally be designed and constructed to conform to the data model representing the organisation of interest. This is typically performed by a skilled analyst, and is not particularly flexible once set up.

It is an object of the present invention to provide a database system which overcomes or mitigates some or all of the above disadvantages.

In accordance with one aspect of the present invention, there is provided a method of operating a database system comprising the steps of:

assigning to every entity, every attribute and every entity occurrence a unique, multi-character expression, the expression having a predetermined hierarchical structure which defines the relationship between each entity, attribute and entity occurrence with every other entity, attribute and entity occurrence in the database and

storing said expressions in an expression set table linking each element of each expression with a natural language meaning relating the expression to a hierarchical level and a position in a data model.

In accordance with a further aspect, the present invention provides apparatus for storing data in the form of a database and for allowing multiple route access to selected portions of the data including:

means for storing, for every entity, every attribute and every entity occurrence a unique, multi-character expression, the expression having a predetermined hierarchical structure which defines the relationship between each entity, attribute and entity occurrence with every other entity, attribute and entity occurrence in the database;

means for storing said expressions in an expression set table linking each element of each expression with a natural language phrase relating the expression to a hierarchical level and a position in a data model;

query formulation means for deriving a source multi-character expression including deterministic and non-deterministic characters which correspond to the query;

comparator means for matching all deterministic characters in the source expression with corresponding character positions in selected target expressions stored in a predetermined target table in the database;

output means for generating an output of all target table entries in which a deterministic source character matches a deterministic target character or in which a deterministic source character matches a non-deterministic target character.

The present invention will now be described by way of example, and with reference to the accompanying drawings in which:

Figure 1 shows an exemplary data model useful in describing the present invention;

Figure 2 shows a root expression and extension expression in accordance with those used in the present invention;

Figure 3 shows a symbolic portion of a data model useful in explaining aspects of the present invention;

Figure 4 shows a pair of expressions to illustrate context switch links therebetween;

Figure 5 a plurality of table structures and their inter-relationship which can be used in the implementation of the present invention; and

Figure 6a to 6d show portions of exemplary entity history tables.

The data model

In the present invention, the physical model, ie. the storage model which represents the physical structure of the data stored on the computer system is designed to be much closer to a conceptual model of the real world, ie. the data model of the organisation(s) using the database. This closeness is normally difficult to achieve, simply because the requirements of the computer-accessed

disks and other storage media are so different from the human view of the organisational structure being represented by the database. A database implementation which can simplify the interface between the physical model and the conceptual model offers huge advantages in terms of the speed of processing when accessing information from the database, and also greatly simplifies the software and hardware interface necessary to achieve this interface.

In the present invention, every entity, every attribute and every occurrence of every entity in the data model is uniquely specified by a multi-character "expression" which is conveniently (for the sake of clarity of explanation) divided into a number of "words". In a presently preferred embodiment, the "expression" comprises three five-byte words, with each byte representing one ASCII character selected from a set of approximately 200.

The expressions do more than simply provide a unique label to each entity, each attribute and each occurrence of each entity, but also implicitly encode the data model by reference to its hierarchical structure and protocol. This is achieved by use of the strict hierarchical protocol in the assignment of expressions to each entity. This can be achieved automatically by the database management system when the user is initially setting up the database, or preferably is imposed by a higher authority to enable the database structure to conform to wider standards thereby ensuring compatibility with other users of similar database systems.

The way in which the database structure is imposed by the assignment of these expressions is best described with reference to an exemplary data model as shown in figure 1.

The tree structure in figure 1 represents the "known universe" of the data model. Each hierarchical level of the data model is shown horizontally across the tree structure, and each one of these hierarchical levels may be represented by an appropriate byte I_1 to I_{15} of the expression shown vertically on the left hand side of the drawing. At the highest level of the tree I_1, we have context information defining the organisation using the data, for example the National Health Service, Prison Service, Local Authority, Educational Establishment etc.

The significance of byte I_2 will be discussed later, but broadly speaking indicates a data type from a plurality of possible data types which might be used in one implementation of the present invention.

Within each organisation (eg. the Health Service) there may typically be a number of departments or functions or data view types (represented by byte I_3) such as administration, finance / accounts and clinical staff, all of whom have different data requirements. These different data requirements encompass:

a) different data structures or models reflecting different organisational hierarchies within the department;
b) different views of the same entities and occurrences of entities; and
c) the same or different views of "standard format" data relating to different occurrences of similar or identical entities or attributes.

The significance of this to the present invention will become clear as one progresses downward through the hierarchy.

Each department may wish to segregate activities (eg. for the purpose of data collection and analysis) to various regional parts of the organisation: eg. a geographically administered area or a sub-department. This can be reflected

by expression byte I_4. Each geographically administered area may further be characterized by a number of individual unit types, such as: (i) hospitals, health centres etc. in the case of an NHS application; (ii) schools or higher education institutions in the case of an education application; (iii) prisons and remand centres in the case of the prison service application.

Each of the organisations and units above will have different data structure requirements (as in (a) above) reflecting different entities, attributes and entity relationships within the organisation and these are provided for by suitable allocation of codes within the I_6 to I_{10} range of expression bytes. In this case, the same alphanumeric codes in bytes I_6 to I_{10} will have different meaning when in a branch of the tree under NHS than when under, eg. the education branch, even though they exist at the same hierarchical level. As an example, the sub-tree structure represented by particular values of bytes I_6 to I_{10} may refer to patient treatment records in the NHS context, whereas those values of codes may refer to pupil academic records in the education context.

However, in the case of (b) above, where the organisational unit requires the same or different views of the same entities, attributes and occurrences of entities as other organisational units, the codes in bytes I_6 to I_{10} of one branch of the tree will represent the same underlying structure and have the same meaning as corresponding byte values under another branch of the tree. An example of this is where both the administration departments and the finance departments require a view of the personal details of the staff in the hospital, both doctors and nurses. Note that the views of the data may be the same or different for each department, because the view specification is inferred from the higher level I_1 to I_5 fields. In this case, as will be explained later, for entities, attributes and occurrences of entities which are the same in each sub-branch, some or all of the codes I_{11} to I_{15} which identify each entity occurrence will have identical values.

In the case of (c) above, ie. the same or different views of standard format data relating to different occurrences of similar or identical entities and their attributes, it will be understood that a number of predefined bytes require the same specification regardless of the particular organisation using them. For example, a sub-tree relating to personnel records, and including a standard format data structure for recording personnel names, addresses, National Insurance numbers, sex, date of birth, nationality etc. can be replicated for each branch of the tree in which it is required. For example, all of the organisations in the tree will probably require such an employee data sub-tree, and thus by use of standardized codes in bytes I_6 to I_{10} such organisational sub-trees are effectively copied into different parts of the tree. However, in this case, the context information in fields I_1 to I_5 will indicate that within each organisation, we are actually dealing with different occurrences of similar format data.

In preferred embodiments of the present invention, the tree structure defined by the expressions I_1 to I_{15} is used to define not only all entity types, all entity attribute types and all entity occurrences, but is also used to encode the actual attribute values of each entity occurrence where such values are limited to a discrete number of possible values. For example, in the sub-tree relating to treatments in the NHS hospital context, "drug" is an entity which has a relation with or is an attribute of, for example: doctors (from the point of view of treatments prescribed); patients (from the point of view of treatments given); administration (from the point of view of maintaining stocks of drugs) and so on. The entire set of drugs used can be provided for with an expression to identify each drug. In a preferred embodiment, the parts of the expression specific to the occurrences of each drug will be located in the I_{11} to I_{15} fields as shown in figure 1. Thus when used in conjunction with the appropriate fields I_1 to I_{10}, it will be apparent whether the specified drug is in the context of a treatment prescribed by a doctor, a treatment received by a patient, or a stock to be held in the hospital pharmacy.

Further bytes in the expression, lower in the hierarchy can be associated with the drug to describe, for example, quantities or standard prescription types. It will be apparent whether the expression refers to a prescribed quantity or a stock quantity by reference to the context information found higher in the hierarchy. In practice, the number of discrete values allowed for each of these grouped "entity values" using the five fields I_{11} to I_{15} is approximately $200^5 = 3.2\times10^{11}$. As will be described later, the number of permutations allowed can actually be expanded indefinitely, but in practice this has not been found to be necessary.

Thus, in the fifteen character expression I_1 to I_{15}, each character represents a English language expression defining some aspect of the data model, and by travelling downward through the table it is possible to compose a collection of English expressions which represents the complete specification of an entity, an attribute or an entity occurrence.

Implementation

For the following detailed description of an implementation of the present invention, we shall use the following data modelling scheme, although it will be understood that the method of the present invention applies to a wide variety of data model designs. In this data model, all information may be regarded as being about either a "presentation", or an "activity". A presentation is some thing or notion which is presented to the user. An activity is some thing or notion which is initiated by the user.

For example, the tree structure of figure 1 is, largely, a presentation to the user about the hierarchical structure of the organisation in which the user is working, and the lists of patients, doctors, nurses, school pupils and

prisoners are all presentations of things in existence and their relationship with that data structure.

Activities may be regarded as events which the user records or initiates which affect the database - ie. treating a patient, updating a person's records, ordering further supplies of drugs etc. An activity is often initiated in response to a presentation - that is to say, the doctor may view the relevant records of the database presented to him in the form of patient treatments, and prescribe further treatment based thereon.

Information about presentations and activities can only be recorded when associated with an object. In other words, the "patient treatment" per se has only abstract meaning until it is linked with a particular patient, doctor, hospital etc.

The exemplary database structure will be described with reference to five activities which take place in the creation and maintenance of the database. "Registration" we describe as the recording of static information about an object's existence, which is all presentation information. The registration process itself can, however, be regarded as an activity. This process is embodied in the steps of constructing the tree structure of figure 1, and recording information about each entity at the "bottom" layer of the tree: ie. identifying drug no. 1 as "aspirin".

"Profiling" we describe as recording information about the object's condition, which information is likely to change, and may therefore be regarded as dynamic. The distinction between static and dynamic information is not a rigid one: static information can also be subject to change, and dynamic information might not actually change. An example of dynamic information might be the assignation of a patient to a certain doctor for a certain type of

treatment. Compare this with static information represented by the recording of the existence of the patient in the data model by giving the patient a unique identifying number. Loosely speaking, the registration of static information is the identification of each entity occurrence at the bottom of the tree structure of figure 1 (eg. DrugNo1, DrugNo2,... DoctorID1, DoctorID2,... PatientID1, PatientID2...) and the profiling activity corresponds to defining an entity occurrence's (PatientID1) relationship with the tree (eg. associating the patient with HospitalNo1 or DoctorID1).

"Response" is regarded as recording information about responses to an object's condition - ie. updating patient records with treatment details etc.

"Event logging" is regarded as recording information about a sequence of events associated with responses to an object's condition. This is activity and presentation information. For example, it is necessary to ensure that the history of a patient within the hospital can be tracked over a period of time, with details of all treatments and referrals indicated.

"Reporting" allows the user of the database to query the system to extract specified information therefrom.

To carry out all of these activities, a database engine uses the expression set introduced above. As previously discussed in general terms, a full expression consists of, in a presently preferred embodiment, fifteen elements, divided into three groups of five elements, also known as words. The first word, I_1 to I_5, is reserved for context specification, ie. whose view the expression reflects, the sense of the expression and the domain of the expression. We shall call this the context word. The second word, I_6 to I_{10}, is reserved for specifying a particular procedure, entity of event in the context specified by the context word. We shall call this the specification word. The

third word, I_{10} to I_{15}, is reserved for specifying qualitative information regarding the procedure, entity or event. We shall call this the qualitative word.

Thus, in addition to defining the upper layers of the tree structure shown in and described with reference to figure 1, the context word can also specify the sense of the data identified by the expression. For example, there may be codes embedded into the context word which indicate that we are dealing with presentation information, activity information, or response information, etc. In the preferred embodiment described with reference to figure 1, this "sense of the data" is indicated by the value of byte I_2.

The expression can thus be a complete description of a situation: a place, an associated event and a frequency of occurrence; or perhaps a person, an associated action and some measure of quality of that action.

This is in contrast to other coding systems which are usually of an atomic nature. In atomic coding systems, a particular code will describe a particular feature, action or state. To fully describe a situation, then, an arbitrary number of codes must be grouped together in some way. For example, one code for a place, another for an event, perhaps another to complete the event description and then a qualifier of some sort. There is nothing in the codes to indicate the relationship these codes have with one another.

The expressions used in accordance with the present invention have a grammar. Each expression indicates a context, a specification and a quality. If any of these components are unknown or irrelevant, then by default, the expression indicates that this is so by the use of "wild card" characters (which we shall generally refer to as non-deterministic characters).

We also note, at this stage, that although the embodiment described here uses an expression which is fifteen characters in length, to represent a tree structure that is fifteen layers deep, extension trees are possible. The expression may include a unique code in the third word bytes I_{11} to I_{15} which indicates that the word represents a pointer to a further expression. For example, with reference to figure 2 there is shown an exemplary extension expression. Root expression 200 contains the three five-character words, with the final five characters representing the link to extension expression 201. The third word 202 includes a special designated character (shown as "X") in position I_{11} which indicates that the following four characters in bytes I_{12} to I_{15} represent a pointer label to the extension expression 201. The pointer label is replicated in the first word 203 of the extension expression. Thus the presence of the "X" in byte I_1 identifies the status of the expression as an extension expression. In the extension expression, the characters in bytes I_6 to I_{15} represent a further sub-tree appended to the main tree of figure 1. It will be understood that extension expressions may be used to greatly increase the number of entity occurrences, attributes or ranges of possible values of attributes over that which is provided by the root expressions.

A blank element in an expression is used to indicate that there is no further detail in an expression. Thus every element to the right of a blank element must also be blank. This can be understood by recognising that a branch of the tree in figure 1 cannot exist unless it is connected to the root via branches hierarchically above it.

Where there is no specification at any position in the expression, this is indicated by the wild card symbol "#".

A feature of the use of expressions to describe the data model is that similar data structures, or sub-trees, are replicated throughout the main tree by

using similar expression patterns. For example, with reference to figure 3, sub-trees 301, 302 have the same structure, and sub-trees 303, 304 have the same structure.

There are three paths down through the hierarchical tree of figure 3. With traditional hierarchical browsing systems, the user would explore their way down the tree to the extremities. This is also the case with the present invention. However, the use of the expression sets also provides the ability to jump to other similar places in the expression set. This can be done in all hierarchical systems by back-tracking until a branch is reached where an alternative route is decided upon. Then the user would explore this new route until the require branch is reached. With reference to figure 3, at least six steps are required to get from path CBBCF to CLBCF. Each step presents the opportunity of making a wrong decision that would delay the finding of the correct data. The use of expressions allows one step jumping from sub-tree 301, or 303 to sub-tree 302, or 304 which would otherwise need many back and forward tracking steps to be made.

This is achieved by the positional integrity, or the "place value" of the characters within the expressions. It can be seen that by changing the B in the second position to an L, the correct expression is arrived at. The lower level elements remain the same which, through the rules of positional integrity, means that the detail description is identical but we may now be talking about a member of staff in the NHS or a member of staff in the prison service.

A further application of this feature is that the data model can be arranged to permit data type context changes to be made by changing perhaps only one higher order digit in the expression. For example, the high order character I_2 is chosen to represent the context of the data model - eg. "presentation" or "response", then, as shown in figure 4 for example, whilst

diagnosing a particular disorder at a detailed level, by changing the value of one high order element in the specification word, the user can be left in perhaps the response region of expression codes for this particular disorder. In the example of figure 4, this is illustrated by changing the second order element in the context word from "G" to "V".

An overview of the use of an expression set together with the implementing tables which comprise a preferred embodiment of the database system of the present invention is now described with reference to figure 5.

Every occurrence of an entity about which information must be stored is recorded in the entity details table 510. Each occurrence of each entity is given a unique identifier 512 which is assigned to that entity occurrence, and information about the entity is stored as a value expression information string 513. Examples of value expressions are the character strings giving names, street addresses, town, county, country etc, or drug name, manufacturer's product code etc. These details are essentially alphanumeric strings which themselves contain no further useful hierarchical information and are treated solely as character strings. As will become apparent later, the decision as to which occurrence values are handled at this level is determined by the user's requirements. For example, an address may be recorded entirely as character strings having no further hierarchical significance. Alternatively, the county or city field, or postcode portion of an address might usefully be encoded into an expression in order that rapid searching and sorting of, for example, geographical distribution of patients becomes possible.

Entering this information may be regarded as a registration activity, in that static information about an object's existence is being recorded in the database.

Attributes which may only take permitted discrete values from a set of possible values are effectively recorded in the expression I_1 to I_{15} associated therewith as will be described later.

The unique identifier 512 of each entity occurrence in the entity details table 510 provides a link to an entity history table 520 where entry of, or update to the entity occurrence status is stored. In this table, the event updating the database is given a date and/or time 524, an expression 526, and the unique identifier 522 to which the record pertains, and may include other information such as the user ID 527 of the person making the change.

This activity is "profiling": in other words recording information about the entity and its relationship with the data model. An example of this is assigning to PatientID1 (from the entity details table 510) an attribute value, HospitalNo1 by use of the appropriate byte I_5 in the expression.

In the entity history table 520, various details of the event being recorded may not be available, or may have no relevance at that time. For example, a new patient in a designated hospital may be admitted, and some details put on record, but the patient is not assigned to any particular doctor or ward until a later time. Additionally, some information may be recorded which is completely independent of the user view or other context information. Thus the event is logged with only relevant bytes of the expression encoded. Bytes for which the information is not known, or which are irrelevant to the event are non-deterministic and are filled with the wild card character, "#".

The entity history table 520 may also include an event tag field 528 which can be used in conjunction with a corresponding field in an episode management table to be described hereinafter. It will indicate which coding activity was being carried out when the expression was assigned to the entity.

For example, this tag could indicate whether the coding was carried out during an initial assessment, an update, a correction, a re-assessment, etc. This tag also orders entity codes into event groups. For example, in the medical context, when a person enters the system as a patient, they initiate an episode. An episode can have many spells, and a spell can consist of many events. What is more, a patient can be undergoing more than one episode at a time, and under each episode, more than one spell at a time. Many organisations need to store this sort of information for costing and auditing purposes. By coding this information into an expression, it will be possible to browse this information.

The entity history table may also include a link field 529 which is designated to link related groups of codes allocated during a particular entity-event-times. For example, in a social services application, a home visit, a visit date, miles travelled and the visitor could all have an expression associated with the visit event. The link field will link these expressions together. Alternatively, the event tag field may also cater for this function.

A memo field 523 may also be included in the entity history table to allow the user to enter a free text memorandum of any length for each code allocated to an entity. In effect, every time a field is filled, a memo can be added.

The expression set of the entire database is recorded in a third table, the expression set table 530. This encodes each expression against its English language meaning, and effectively records the data model as defined by the hierarchical structure of figure 1. There is an English language meaning for each byte of the expression, each byte representing a node position in the data model tree, and the precise significance of every occurrence of every entity or attribute is provided by concatenating all English language meanings for each

byte of the expression: eg. NHS - Presentation Data Type - Administrator's View - Region1 - HospitalNo2 - Doctor Record - Name - DoctorID1.

As has been discussed previously, the expressions may include expression extensions which map a sub-tree onto the main tree. For convenience, these extension expressions can be located within the expression set table 530 (the extension entries being identified by the byte I_1, or could be located in a supplementary table (not shown), in which the pointer fields I_{11} to I_{15} of the main expression are used as the first fields I_1 to I_5 of the extension expression.

Further details of the tables and their structures will be discussed hereinafter. In use, the database management system first constructs the data model tree structure in the expression set table 530, with each expression being allocated a corresponding English term. This can be done by dialogue with the user, or by systems analysis by an expert. Preferably, the use of pre-formatted codes representing certain data structures are used by many different users. For example, personnel file type structures may be used by many different organisations. This allows compatibility of databases to allow data sharing between organisations, with users being allocated blocks of codes for their own user-specific purposes, as well as using shared codes which have already been defined by a higher authority.

In figure 6a, an exemplary expression set table portion 600 representing a personal details sub-tree is shown. It will be observed that fields I_8 and I_9 represent the personal detail sub-tree data structure which can be replicated for any part of the tree. That is to say, the sub-tree 601 can represent attributes of a patient (as shown) or in a different part of the tree may represent attributes of a prisoner, or member of staff. Note that the "names" grouping 602 (I_8 = "1") provides a sub-tree of entity attributes, eg. "surname", "first name" etc.,

each of which will have a number of entity occurrences associated therewith, each having specific values. Each occurrence will be separately identified using the lower order fields of the expression (not shown). The actual values will be installed as character strings in the entity details table 510 (see figure 5). By contrast, the country of origin entity 603 (I_8 = "5") provides a sub-tree of discrete entity attribute values: eg. "England", "Scotland", "Wales", "Belgium", "France" etc. Thus, in this case, the tree structure (ie. the expression itself) can provide the individual attribute values of the entity "Country of Origin".

In figure 6b, a further exemplary expression set table portion indicates a sub-tree relating to diagnoses on a patient. Only the expression values I_{11} to I_{15} are shown for brevity. This expression sub-set provides a sub-tree of possible attribute values relating to diagnoses or operations etc. In figure 6c we show an expression set table portion representing a sub-tree which can be used to provide a "standard" range of discrete attribute values relating to angles between 0 and 180 degrees. In figure 6d we show an expression set table portion 630 representing categories of medical presentations.

Figures 6a—6c demonstrate a further possible embodiment of the database implementation of the present invention. In figure 6a it is noted that bytes I_{11} to I_{15} are not shown. In practice, there can be some advantages in operation in constructing separate tables to contain discrete "chunks" of the expression set table 530, that is chunks relating to adjacent groups of I_{11} to I_{15} codes which all relate to the same I_1 to I_{10} value. These each form an extension table 540 which is pointed to by an extension table pointer located in column 606 of the expression set table 600. This is particularly useful where repeating chunks of I_{11} to I_{15} codes are found in many places throughout the expression set table 530. Thus sub-tables 610, 620, 630 could be tables in their own right,

not forming part of the main table, and can thus be used at numerous locations down the main table 530.

In figures 6a - 6d, the expression set table uses a standard notation. Because of the hierarchical nature of the expressions, it is essential to maintain positional integrity. Thus, with reference to figure 6a, the patient sub-tree must commence at level I_8 of the tree structure, regardless of the complexity of the tree structure above. Thus, if there is only one organisation using the database, or if there a limited number of user views required, there may be no requirement to use some higher order context specifiers (eg. I_4 and I_5). These unused fields have no specification at that point and are represented by "#". Where there is no specification this is represented in the English term field 605 by the symbols "< >". In constructing the table, for implementational reasons discussed later, it is highly desirable that the table is maintained in strict alphanumeric order of expressions, with discontinuities between higher and lower tree branches filled in with blank specification lines (ie. those represented by "< >"). It will be understood that these correspond to particular levels within the tree structure for which there are no divisions of branches.

Additional fields may be included in the expression set table. For example, a note flag field 532 may be used to signify that explanatory information is available for a term. This would typically provide a pointer to a notes table. A symbol in this field could indicate the existence of, for example, passive notes (information available on request); advisory notes (displayed when the code is used); and selection notes (displayed to the user instead of the English term). A sub-set field 533 may also be provided for expression maintenance tasks, but these are not discussed further here.

When an expression set table has been constructed, it can be related to individual entity occurrences in the following manner. As previously

discussed, the unique occurrences of entities can be placed in the entity details table 510, each having a unique identifier 512. This is linked to the expression set table, and thus to the tree via the entity history table. This records the entity unique identifier 512 in a column 522 and links this with the appropriate expression or part expression 526. The date of the event is logged in field 524, and other details may be provided - eg. whether the data entry is a first registration of a record, whether it is a response record (eg. updating the database) etc.

In a preferred embodiment, all of the data will be contained in five core tables as already identified. Other tables may exist containing statistical information and derivations of this core data. The purpose of these secondary tables is to hold semi or wholly prepared reporting data for frequently performed queries. The exact number and make-up of these secondary tables depends on reporting and querying requirements. These secondary tables are preferably directly maintained by the database management system so that the end user will gain the benefit of them whilst never seeing them.

The entity ID table 550 (figure 5) is an example of a secondary table which is used when communicating and sharing data with other systems. This table matches the entity unique identifier ID codes with entity ID codes used by other systems.

It is also possible to record static entity details in a form which is structured ready for input and output. For example, name, address and telephone records may be stored in successive columns of an address table 560, each record cross-referenced to the main data structure by the expression code or cross-referenced to an entity by the expression code I_1 to I_{15}. The link can thus be made with either the expression set table 530 or the entity history table 520. Then, whenever that branch of the tree is accessed pertaining to one

individual record, the full "static and demographic details of that entity occurrence may be accessed from a single table.

A similar arrangement is shown for providing detailed drug information, by drug table 570.

A further modification may be made to the embodiments described above in respect of the use of the entity details table 510. It is not essential for all information about an entity occurrence to reside in the entity details table 510. In some models, it is advantageous to restrict the use of the entity details table 510 to that of a "major entity" only — the most significant entity forming part of the modelled organisation. For example, in the hospital environment, the patient could be chosen as the major entity. In this case, all other (non-structural, character-string) information about entities can be located in an appropriate field of either the entity history table 520, or the expression set table 530. In the case of the entity history table 520, an appropriate field to use is the memo field 523, and in the case of the expression set table 530, an appropriate field to use is the English term field 535. It will thus be understood that, where the non-structural information held about even the major entity is small, the entity details table 510 can be dispensed with all together.

Reporting

The present invention, as described above, offers significant advantages in the execution of reporting and database querying functions.

This will be illustrated in two ways. To answer a given query, the database system defines a query expression comprising fifteen bytes which correspond with the expressions as stored in the entity history table 520 and expression set table 530. The query expression will include a number of

deterministic bytes and a number of non-deterministic bytes. The non-deterministic bytes are effectively defined as the wild-card character "#" — "matches anything". The deterministic bytes are defined by the query parameters.

For example, a simple query might be: "How many patients are presently registered at hospital X." To answer this query, the query expression imposes deterministic characters in fields I_1 (=NHS), I_4 (=hospital identity), I_6 (=patients). Other context information may be imposed by placing deterministic characters in bytes I_2 (=presentation information). All other bytes are non-deterministic and are set to "#". The database scans through the expression set table matching the deterministic characters and ignoring others. Note that in the preferred embodiment, the expression set table is maintained in strict alphanumeric sequence and thus very rapid homing in on the correct portions of the database table is provided where high-order bytes are specified. This will normally be the case, since the hierarchical nature of the expression set will be arranged to reflect the needs of the organisation using it. The database system can then readily identify all the tuples of the expression set table providing a match to the query expression.

One of the major advantages of the present invention will now become evident. The answer to the initial query has effectively homed in on one or more discrete portions of the expression set table and counted the number of tuples matching the query expression. Supposing that the user now requires to "progressively browse" by stipulating additional conditions: "How many of those patients are being prescribed drug Y" requires only the substitution of the non-deterministic character "#" with the appropriate character in the requisite field I_n of the expression to change the result. Similarly, carrying out statistical analysis of other parameters, such as: "How many patients were treated by doctor Z with drug Y" can rapidly be assessed. It will be understood that

progressively narrowing the query will eventually result in all bytes of the query expression becoming deterministic and yielding no match, or yielding a single patient entity match whose details can then be determined by reference to the entity details table 510 (or the appropriate memo field).

The key to the speed of result of the statistical querying function is the construction of the expression set table. When imposing conditions on various attributes of an entity, ie. by setting a deterministic character in a byte of the query expression, the relevant data will be found in portions of the table in blocks corresponding to that character. Progressive sub-querying requires only scanning portions of the table already identified by the previously query. Even where a higher level context switch takes place, relevant parts of the expression set table can be accessed rapidly as they appear in blocks which are sequenced by the expression hierarchy.

Scanning the table can be achieved most efficiently by recognising that only the highest order, deterministic byte of the query expression need be compared with corresponding bytes of each record in the expression set table until a first match is obtained. Thereafter, the next highest order byte must be included, an so on until all deterministic bytes are compared. This results from maintaining a strict alphanumeric ordering to the table.

A second type of querying relates to examining the historical aspects of the database. For example, the query may be, "In the last year, what drugs and quantities have been prescribed by doctor X?" To answer this query, the query expression is formulated in the same manner as before, imposing deterministic bytes in the appropriate places in the query expression. This will include one or more "lowest order" bytes in I_{11} to I_{15} which actually identify a doctor, and non-deterministic characters against the drug fields. This time, however, the entity history table 520 is scanned, in similar manner, seeking

only matches of deterministic characters. In a preferred embodiment, the entity history table will be maintained in chronological sequence and thus the search can be limited to a portion of the table where date limitations are known and relevant. Matches of deterministic characters will be found throughout the table where a relevant event relating to prescription of a drug by doctor X is found. Note that the entity history table may include other fields which can be used to impose conditions on the query, such as the user ID of the person entering the record.

A third type of querying relates to analysis of the records pertaining to a single entity value: the entire medical record of patient X. In the preferred embodiment, patient X would be identifiable from the entity details table 510. The query would initially involve searching for the patient's name to locate the unique identifier (unless that was already known). Once the unique identifier for a patient was known, then the entire entity history table can be scanned very rapidly for any entry including the unique identifier. The strengths of the present invention will then be realized in that the output from this scan will provide a number of entries each of which carries all of the relevant information about that patient incorporated into the extracted expression bytes I_1 to I_{15}. The entire patient's record can then be "progressively browsed" without recourse to any further searching operation on the main entity history table. Specific details of the patient's treatments, doctors, hospital admissions, prescriptions etc are all very rapidly available at will be assertion of appropriate deterministic bytes in the expression I_1 to I_{15}.

It is noted that the event history table will include many records where the expression stored in the record contains many non-deterministic bytes. For example, where a doctor X prescribes a patient Y with drug Z, other bytes of the expression may be either not known, or not relevant. For example, the patient may have been assigned to a ward W in the hospital which could be

identified by another byte. However, this venue in which the treatment took place might be: a) unknown; b) known but not relevant to the record; or c) automatically inferrable from the context of the person making the record entry. Whether this information is included in the record is stipulated by the users; however, it will be noted that it does not affect the result of the query whether the byte in the entity history table relating to WARD W is deterministic or non-deterministic, because the query expression will set that relevant byte to non-deterministic unless it is stipulated as part of the query.

When the database system has extracted all of the records of the entity history table matching the query expression, it preferably saves these to a results table for further querying, or progressive browsing. For example, the results table can then be analysed to identify which treatments were made at an individual hospital, or by an individual doctor by setting additional conditions on particular bytes of the query expression. Memo fields can be extracted to view comments made at the time of treatment. It can be seen that the results table formed in response to the initial query actually contains all of the information relevant to a given patient's treatment, and not just the answer to the initial query "What drugs have been prescribed to patient X?".

In summary, then, the method of the present invention requires storing the information of the database in such a manner that data for a query may be extracted far more rapidly than existing prior art database storage schemas, and with an expression for each extracted record. The presence of this expression in the query result has an important effect. A unique reporting benefit gained is the scope for progressive browsing and "interactive reporting". When a database query is executed to provide information for a report, the answer will be made up of a number of expression records. This subset of expressions inherits all the structural information held in the main expression set.

As a general example: a detailed report on the number of severe hallucination instances in a given geographical area during the past year might return a subset of 12,000 expressions. Because these are full expressions, higher and lower level information is also inherent in this subset. Further investigation of the answer through browsing the returned hierarchy might reveal that 70% of cases were male, or 30% of cases occurred in the prison service, etc. Similarly, a high level report on the number of instances of hallucination in a particular organisation might return a subset of 9,000. More detailed information will be inherent in this retrieved subset. By progressive browsing of this subset, it may transpire that 90% of mild occurrences were in planning departments or that 5% of severe occurrences were in education departments. The processing time required to browse this information with further, more detailed, "sub-queries" is substantially speeded up over prior art systems simply because the expression set readily provides all the lower level information.

A further advantage of the present invention is provided by the link between the expressions 531 in expression table 530, and the English language terms 535. By retrieving information as expression sets, all the information implicit in the hierarchy is passed along with its inherent structure. There is a one to one relationship between each expression, and its implicit meaning in terms of the whole data model is defined in the English language term. By providing copies of the expression set table with different terms 535 defined, an expression can be decoded into any language or effectively mapped onto any modified data model which has the same hierarchical structure as the original.

Thus a French speaking person may query the English language database by provision of an alternative expression set table 530 with the appropriate French terms replacing the English terms 535. This result is possible because

all language dependence of the data is divorced from the data structure (expression set) and the data access mechanism to that data structure.

The present invention can be readily realized both in software, and in hardware. It will be understood that the database querying essentially requires rapid fifteen byte wide comparison of the expressions I_1 to I_{15}. An extremely fast co-processor ASIC could thus be manufactured which includes up to fifteen eight-bit comparators in parallel. In practice, querying would never require all fifteen bytes to be compared, as most queries involve the setting of a large number of the bytes to a non-deterministic state, thus in practice requiring fewer parallel circuits and enabling simplification of the design of a dedicated co-processor.

CLAIMS

1. A method of operating a database system comprising the steps of:

assigning to every entity, every attribute and every entity occurrence a unique, multi-character expression, the expression having a predetermined hierarchical structure which defines the relationship between each entity, attribute and entity occurrence with every other entity, attribute and entity occurrence in the database and

storing said expressions in an expression set table linking each element of each expression with a natural language phrase relating the expression to a hierarchical level and a position in a data model.

2. A method according to claim 1 further including the step of storing a plurality of entity occurrence values in an entity details table and linking each entity occurrence with at least one respective expression.

3. A method according to claim 2 wherein the respective expression defines an attribute value of the respective entity occurrence.

4. A method according to claim 2 wherein the respective expression defines a position of the respective entity occurrence in the data model.

5. A method according to claim 1 further including the step of storing, in an entity history table, a plurality of records, each record including the value of at least one attribute of an entity occurrence as defined in accordance with the data model which is defined in the expression set table.

6. A method according to claim 5 further including the step of, for each record in the entity history table, including a field indicating the chronology of the record with respect to other records.

7. A method according to claim 6 further including the step of updating the database by adding further records to the entity history table.

8. A method according to claim 5 wherein the expression includes predetermined characters representing a non-deterministic value of an attribute.

9. A method according to any one of claims 5 to 8 further including querying the database to determine the status of an entity, or class of entities by the steps of:

for a given set of query parameters, defining the characters of the expression which are deterministic to the query and those which are not deterministic to the query to define a query expression containing deterministic and non-deterministic characters;

scanning at least a selected portion of the entity history table to examine the expression contained in each record;

matching every deterministic character of the query expression with every deterministic character in the examined record; and

where each deterministic character of the query expression matches the respective record expression, extracting the record to a results table.

10. A method according to claim 1 wherein the entries in the expression set table are stored in a predetermined alphanumeric sequence.

11. A method according to claim 1 wherein the entries in the expression set table include a high order context character indicating the type of activity during which the data was recorded on the database.

12. Apparatus for storing data in the form of a database and for allowing multiple route access to selected portions of the data including:

means for storing, for every entity, every attribute and every entity occurrence a unique, multi-character expression, the expression having a predetermined hierarchical structure which defines the relationship between each entity, attribute and entity occurrence with every other entity, attribute and entity occurrence in the database;

means for storing said expressions in an expression set table linking each element of each expression with a natural language phrase relating the expression to a hierarchical level and a position in a data model;

query formulation means for deriving a source multi-character expression including deterministic and non-deterministic characters which correspond to the query;

comparator means for matching all deterministic characters in the source expression with corresponding character positions in selected target expressions stored in a predetermined target table in the database;

output means for generating an output of all target table entries in which a deterministic source character matches a deterministic target character or in which a deterministic source character matches a non-deterministic target character.

The Patent Office

32

Application No: GB 9419719.1
Claims searched: 1-12

Examiner: Mr S J Probert
Date of search: 11 September 1995

Patents Act 1977
Amended Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.N): G4A AUDB

Int Cl (Ed.6): G06F 17/30

Other: Online : WPI, INSPEC

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A	EP 0638870 A1	(Hitachi, Ltd.)	
A	EP 0583108 A2	(Automated Technology Associates)	
A	US 4479196	(Ferrer et al)	

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.